REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1996	Final, Sept. 1993 - Aug. 1996

4. TITLE AND SUBTITLE

RSTA on the Move: Detection and Tracking of Moving Objects from an Autonomous Mobile Platform

5. FUNDING NUMBERS

DAAH04-93-G-0419

6. AUTHOR(S)

Larry S. Davis, Carlos Morimoto, Martin Herman, Ruzena Bajcsy and Randal Nelson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3275

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 N. Fairfax Dr., Arlington, VA 22203-1714
Army Research Office, P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

ARO 32365.11-MA

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release.
Distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report describes accomplishments on a UGV RSTA project conducted by a consortium led by the University of Maryland and including the University of Pennsylvania, the University of Rochester, and the National Institute of Standards and Technology. We first review work done on the design, implementation and integration of real-time vision algorithms for image stabilization, detection of moving objects from a moving platform and camera control. We then present brief descriptions of a number of supporting basic research projects conducted by the members of the consortium.

14. SUBJECT TERMS

Unmanned ground vehicles, Reconnaissance/surveillance/target acquisition, Image stabilization, Moving object detection, Camera control

15. NUMBER OF PAGES

52

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements.***

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

RSTA on the Move: Detection and Tracking of Moving Objects from an Autonomous Mobile Platform

Final Report
September 1993 – August 1996

Author: Larry S. Davis

October 31, 1996

U. S. Army Research Office

Grant DAAH04-93-G-0419
ARPA Order No. A422

Institution: Center for Automation Research
University of Maryland
College Park, MD 20742-3275

CONTENTS

LIST OF FIGURES

LIST OF TABLES

FOREWORD

This report describes accomplishments on a UGV RSTA project conducted by a consortium led by the University of Maryland and including the University of Pennsylvania, the University of Rochester, and the National Institute of Standards and Technology. We first review work done on the design, implementation and integration of real-time vision algorithms for image stabilization, detection of moving objects from a moving platform and camera control. We then present brief descriptions of a number of supporting basic research projects conducted by the members of the consortium.

1 Introduction

Our **RSTA on the Move** project is a program combining

- development and integration activities ultimately leading to an experimental, real-time active vision system for locating and tracking moving targets from a mobile platform, and
- basic research on fundamental active vision problems including motion estimation, image sequence stabilization, detection and characterization of independent motion patterns, and real-time sensor control.

Our integration activities involve algorithm development and integration on the Datacube real-time image processing platform, and experimentation on video sequences obtained, initially, off-line from a sensor mounted on a HMMWV at NIST, and, ultimately, on-line using the same NIST platform with onboard real-time and parallel processing. Section 2 describes our accomplishments on development and integration.

Real-time algorithms for image stabilization and moving object detection have been developed at Maryland, while Rochester has developed a more general real-time algorithm for detection of independently moving objects. Both the Maryland stabilization algorithm and the Rochester independent motion detector have been transferred to the NIST Datacube, and were demonstrated during 1995 at Martin Denver. Other work involves the integration of these two algorithms on a common Datacube/SPARC platform, and design and implementation of spatio-temporal grouping algorithms for focusing attention of the active vision component of our system on an image window containing an independently moving object. Research at the University of Pennsylvania has emphasized camera control algorithms that allow us to track the moving target and to maintain as large an image of the target as possible through control of a zoom lens. Some core camera control software has been ported to the NIST platform, with NIST and Pennsylvania collaborating on the design and implementation of the full camera control subsystem. Figure 1 shows the tasks of the individual contractors.

Overall System Organization

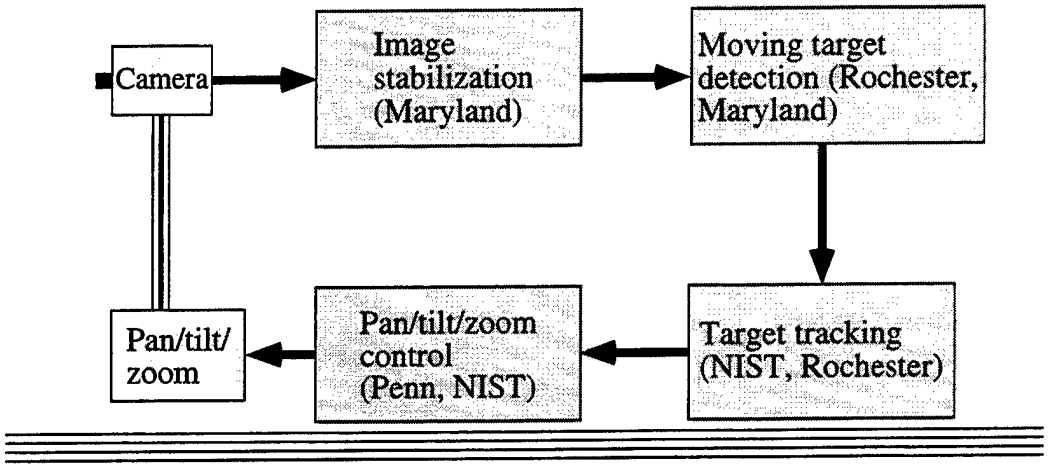


Figure 1: Block diagram of system architecture.

In addition to development and integration activities, the consortium supported a broad spectrum of fundamental research activities in time-varying image analysis and active vision. A set of research vignettes is presented in Section 3, including descriptions of research projects on motion estimation, image stabilization, comparison of image stabilization algorithms in the context of a real-time target acquisition and tracking system (joint research with the Army Research Laboratory), and camera control.

2 Integration Activities

2.1 University of Maryland

(Yiannis Aloimonos, Stephen Balakirsky, Rama Chellappa, Loong Fah Cheong, Cornelia Fermüller, Carlos Morimoto, Yi-Sheng Yao)

Research at Maryland emphasized image stabilization, with some supporting research on detection of constrained (vehicle-like) independent motion. The goal of our image stabilization process was to maintain a stable scene background in a video image sequence. This is accomplished by estimating and compensating for the effects of the movement of the vehicle on the original input image sequence, so that in the resulting stabilized sequence the

background of the scene appears, ideally, as if the vehicle were stationary.

After the sequence is stabilized, independently moving objects can be detected using either the flow-based approach developed at the University of Rochester, or a frame-differencing approach developed by the University of Maryland. The Maryland approach is based on an efficient algorithm for computing a temporal median filter from the stabilized sequence. To optimize for detection of independent vehicle motion, we employ a filtering approach that integrates the results of velocity-tuned filters over several frames and produces the final output of the system. Details of the Maryland work are given below, with an emphasis on the real-time implementation of image stabilization developed by Carlos Morimoto.

2.1.1 Real-time Digital Image Stabilization

Our image stabilization system is composed of two modules: one for motion estimation and a second one for motion compensation. The motion compensation process removes unwanted motion from the desirable smooth motion following the initial estimation process. An advantage of electronic image stabilizers over mechanical ones is that motion can be compensated on demand, offering great flexibility by simply modifying some parameters of the compensation algorithm.

2.1.2 Motion Estimation

Our first prototype described in [9] was based on a multi-resolution image registration technique developed by Zheng and Chellappa [20]. The technique matches a small set of feature points between two frames using a weighted correlation scheme and estimates four affine motion parameters using the computed feature displacements. The processing of each higher resolution level refines the estimates obtained from the registration of the lower resolution levels. This scheme was later replaced by a faster multi-resolution tracking scheme that works from coarse to fine levels. We also adopted a different similarity measure, given by the sum of squared differences (SSD) of local patches around each feature. No estimation or refinement is done between different resolution levels, and the final displacements are used to estimate the motion parameters.

In order to estimate the movement of the camera in a rigid environment we use the model described in [20]. To better understand the model, consider a camera mounted on a balloon. The camera is pointing down at a flat surface S, with its optical axis always perpendicular to S. In such a configuration, translations and rotations of the balloon will cause the image to translate and rotate correspondingly. The image will change scale according to the balloon's changes of altitude.

Under this motion model, the transformation between two different camera positions is defined by

$$\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} = \begin{pmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} \tag{1}$$

where (x_i, y_i, z_i) are 3-D coordinates relative to a coordinate system fixed on the camera at time t_i, for $i = \{0, 1\}$. $(\Delta x, \Delta y, \Delta z)$ define the translation of the camera measured from the coordinate frame at time t_0, and Θ is the rotation angle, also measured with respect to the coordinate frame at time t_0. Image coordinates at time t_i will be denoted by (X_i, Y_i). The image plane defined by the X_i and Y_i axes, which are respectively parallel to the x_i and y_i axes and consequently perpendicular to the z_i axis, has its origin at $z_i = f$ where f is the camera focal length. Under this configuration, the equations for perspective projection are

$$\begin{pmatrix} X_i \\ Y_i \end{pmatrix} = \frac{f}{z_i} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{2}$$

Combining (1) and (2), we get

$$X_0 = \frac{f\, x_0}{z_0} = \frac{f\,(x_1 \cos\theta - y_1 \sin\theta + \Delta x)}{z_1 + \Delta z} = \frac{z_1}{z_1 + \Delta z}\left(X_1 \cos\Theta - Y_1 \sin\Theta + f\,\frac{\Delta x}{z_1}\right) \tag{3}$$

$$Y_0 = \frac{f\, y_0}{z_0} = \frac{f\,(x_1 \sin\theta + y_1 \cos\theta + \Delta y)}{z_1 + \Delta z} = \frac{z_1}{z_1 + \Delta z}\left(X_1 \sin\Theta + Y_1 \cos\Theta + f\,\frac{\Delta y}{z_1}\right) \tag{4}$$

The term $z_1/(z_1 + \Delta z) = z_1/z_0$ is the change in scale and will be denoted by $\mathcal{S}$, while the terms $(f\,\Delta x)/(z_1 + \Delta z)$ and $(f\,\Delta y)(z_1 + \Delta z)$ correspond to the translation in the image

coordinate system, which will be denoted by $(\Delta X, \Delta Y)$. In matrix notation, the relation between two image frames taken at times t_0, t_1 is

$$\begin{pmatrix} X_0 \\ Y_0 \end{pmatrix} = \mathcal{S} \begin{pmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{pmatrix} \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} + \begin{pmatrix} \Delta X \\ \Delta Y \end{pmatrix} \tag{5}$$

Notice that the scale factor $\mathcal{S}$ in this model is inversely proportional to the ratio of the distances between two arbitrary image points at times t_0, t_1. Thus $\mathcal{S}$ can also be obtained by considering any two points in the image frames taken at t_0 and t_1. Let A_0 and B_0 be two arbitrary points in the image frame at t_0 (f_0), and A_1 and B_1 be their corresponding points in f_1. Let δ_i be the distances between A_i and B_i, for $i = \{0, 1\}$. Then the scale factor can be computed by

$$\mathcal{S} = \frac{z_1}{z_0} = \frac{\delta_0}{\delta_1} \tag{6}$$

The next section shows how to use this model to recover the four parameters that describe the 2D motion of the camera. It is assumed for the moment that a set of matched pairs of features is available. Descriptions of the feature detection and tracking process will be given later.

2.1.3 Recovering the Motion Parameters

Given a set S_i of N feature points in frame f_i and the set S_j of corresponding points in frame f_j, the scaling factor $\mathcal{S}$ is computed from (6) since it is invariant to translation and rotation. The distances are taken with respect to the center of mass of each set. The position of the center is given by

$$\overline{X}_f = \frac{1}{N} \sum_{k=1}^{N} X_{fk}\,, \qquad \overline{Y}_f = \frac{1}{N} \sum_{k=1}^{N} Y_{fk} \tag{7}$$

where $(\overline{X}_f, \overline{Y}_f)$ are the coordinates of the center of mass of the feature set S_f and (X_{fk}, Y_{fk}) are the (X, Y) coordinates of feature k in frame f. Let δ_{fk} be the distance from feature k to the center of mass for frame f, so the scale between two frames i and j can be computed by

$$\begin{pmatrix} \delta_{i1} \\ \delta_{i2} \\ \vdots \\ \delta_{iN} \end{pmatrix} = \mathcal{S} \begin{pmatrix} \delta_{j1} \\ \delta_{j2} \\ \vdots \\ \delta_{jN} \end{pmatrix} \Rightarrow \mathcal{S} = \left[\begin{pmatrix} \delta_{j1} \\ \delta_{j2} \\ \vdots \\ \delta_{jN} \end{pmatrix}^t \begin{pmatrix} \delta_{j1} \\ \delta_{j2} \\ \vdots \\ \delta_{jN} \end{pmatrix} \right]^{-1} \begin{pmatrix} \delta_{j1} \\ \delta_{j2} \\ \vdots \\ \delta_{jN} \end{pmatrix}^t \begin{pmatrix} \delta_{i1} \\ \delta_{i2} \\ \vdots \\ \delta_{iN} \end{pmatrix}$$

$$\Rightarrow \mathcal{S} = \frac{\sum_{k=1}^{N} \delta_{jk}.\delta_{ik}}{\sum_{k=1}^{N} \delta_{jk}.\delta_{jk}} \tag{8}$$

where X^t denotes the matrix transpose of X.

The computation of the translation and rotation parameters follows the estimation of the scaling factor. Assuming small rotations, the trigonometric terms of (5) can be linearized to obtain

$$\begin{pmatrix} X_{ik} \\ Y_{ik} \end{pmatrix} = \mathcal{S} \begin{pmatrix} 1 & -\Theta \\ \Theta & 1 \end{pmatrix} \begin{pmatrix} X_{jk} \\ Y_{jk} \end{pmatrix} + \begin{pmatrix} \Delta X \\ \Delta Y \end{pmatrix}, \qquad i = \{1, \ldots, N\} \tag{9}$$

Using (9) for all N matched point pairs, a system of $2N$ linear equations in three unknowns (Θ, ΔX, and ΔY) is obtained. Rearranging (9) in the matrix form $\overline{b} = A\overline{x}$, we have

$$\overline{b} = \begin{pmatrix} X_{i1} - \mathcal{S}X_{j1} \\ Y_{i1} - \mathcal{S}Y_{j1} \\ \vdots \\ X_{iN} - \mathcal{S}X_{jN} \\ Y_{iN} - \mathcal{S}Y_{jN} \end{pmatrix}; \qquad A = \begin{pmatrix} -\mathcal{S}Y_{j1} & 1 & 0 \\ \mathcal{S}X_{j1} & 0 & 1 \\ \vdots & & \\ -\mathcal{S}Y_{jN} & 1 & 0 \\ \mathcal{S}X_{jN} & 0 & 1 \end{pmatrix}; \qquad \overline{x} = \begin{pmatrix} \Theta \\ \Delta X \\ \Delta Y \end{pmatrix} \tag{10}$$

where $\overline{x}$ can be solved using the normal equation $\overline{x} = (A^tA)^{-1}A^t\overline{b}$.

2.1.4 Automatic Extraction, Selection and Tracking of Feature Points

2D models are appropriate for image sequences of distant scenes where perspective effects can be neglected. Unfortunately this is not true for typical sequences taken from a moving wheeled vehicle where it is common to have objects in the scene passing close to the camera. The presence of close objects violates the 2D motion assumption, but this model can still

be used if we are able to restrict ourselves to image points corresponding to distant scene points.

A heuristic rule for selecting features on the horizon is currently being used. The horizon is a very strong visual cue, present in almost every off-road situation. The first step in feature extraction is to smooth the input image by convolving it with a 5×5 Gaussian kernel operator. Then the smoothed image is convolved again with a 5×5 Laplacian kernel operator. The resulting Laplacian image is also used for feature tracking and also thresholded. The thresholded image is divided into N vertical zones, where N corresponds to the number of features to be tracked. Each zone is searched from top to bottom, and the topmost feature is selected for tracking.

The features selected in frame f_{t-1} can be tracked to frame f_t by a multi-resolution refinement scheme using parameter estimation proposed in [21]. Although this process is able to produce very good estimates of the motion parameters, it is computationally expensive since it uses a weighted correlation scheme to determine the best feature matches. We use another similarity measure which is given by the SSD over local windows (SSD windows) centered at the feature points. This measure is computed over a neighborhood (search window) around the candidate matches in frame f_t, and the point which returns the minimum SSD is selected as the best match.

The SSD between two windows of size $W = (2w+1) \times (2w+1)$ centered at $P_{t-1}(x, y) \in f_{t-1}$ and $P_t(u, v) \in f_t$ is given by

$$SSD = \sum_{ij} [(P_{t-1}(x+i, y+j) - P_t(u+i, v+j)^2], \qquad i, j \in [-w, +w]. \tag{11}$$

The use of Laplacian images for feature tracking also helps to reduce the sizes of the SSD windows. The Laplacian operator enhances the regions around the tracked features and smoothes the regions of constant brightness that in general surround the edges. Since the Laplacian operator is rotation invariant, edges of different orientations are equally enhanced so that the Laplacian images are not affected by possible rotations of the input images.

Tracking is performed in a hierarchical fashion. A Laplacian pyramid $L[t]$ is formed by combining several reduced-resolution Laplacian images of frame f_t. Each level of the pyramid will be denoted by $L[t, l]$, where $L[t, 0]$ is the Laplacian image of the original frame

of size Z (an image of size Z has $Z \times Z$ pixels). The size of an arbitrary pyramid level l is determined by $\frac{Z}{2^l}$. The levels of the pyramids $L[t-1]$ and $L[t]$ are used from coarse to fine resolution, where each new processed level contributes to determining the position of the matching feature.

Assume that the pyramids have $j+1$ levels, and level j has the coarsest resolution. After the selection of features is performed using $L[t-1,0]$, they are scaled down to level $L[t-1,j]$ and used to determine their matches in $L[t,j]$. For a feature $P_{t-1}^j(x,y)$ of $L[t-1,j]$, a search for the minimum SSD is performed over a search window of size $S = (2s+1) \times (2s+1)$ centered at the pixel $P_t^j(x,y)$ of $L[t,j]$. Let $P_t^j(u,v)$ be the selected matching point for $P_{t-1}^j(x,y)$. Notice that the maximum displacement supported by this search is only s pixels. For the next pyramid level, the coordinates of these pairs are scaled up by a factor of 2. For the feature $P_{t-1}^{j-1}(2x,2y)$, the search for the minimum SSD is now performed around the pixel $P_t^{j-1}(2u,2v)$. This process is repeated until the finest resolution level 0 is reached. Notice that since the displacement is doubled after every level, the total displacement that this algorithm can handle can be very large even for small values of s.

2.1.5 Subpixel Matching

After the grid-to-grid matches are obtained from the hierarchical search, displacements with subpixel accuracy can be easily computed for the finest resolution level of the pyramid using a differential method [18, 21]. Subpixel accuracy is necessary to eliminate the quantization error introduced when the images are digitized.

If a feature $P_t^0(u,v)$ has offset $(\delta x, \delta y)$ relative to $P_{t-1}^0(u,v)$ (assume they were tracked and registered so that the translation $(\delta x, \delta y)$ is very small), i.e., $P_t^0(u,v) = P_{t-1}^0(u-\delta x, v-\delta y)$, the frame difference can be expanded as

$$\begin{aligned} d(u,v) &= P_{t-1}^0(u,v) - P_t^0(u,v) = P_{t-1}^0(u,v) - P_{t-1}^0(u-\delta x, v-\delta y) \\ &\approx \frac{\partial P_{t-1}^0(u,v)}{\partial x}\delta x + \frac{\partial P_{t-1}^0(u,v)}{\partial y}\delta y \end{aligned} \tag{12}$$

The terms $\partial P_{t-1}^0(u,v)/\partial x$ and $\partial P_{t-1}^0(u,v)/\partial y$ can be approximated by forward differences, so that for a small neighborhood around $P_{t-1}^0(u,v)$ of size $W = (2w+1) \times (2w+1)$

there will be W simultaneous equations

$$D = G\,\Delta \tag{13}$$

where D is the known difference of intensities vector $(P^0_{t-1}(u,v) - P^0_t(u,v))$, G is the known gradient matrix and Δ is the unknown vector of translations, so that

$$D = \begin{pmatrix} d(-w,-w) \\ \vdots \\ d(w,w) \end{pmatrix};\quad G = \begin{pmatrix} \frac{\partial P^0_{t-1}(-w,-w)}{\partial x} & \frac{\partial P^0_{t-1}(-w,-w)}{\partial y} \\ \vdots & \vdots \\ \frac{\partial P^0_{t-1}(w,w)}{\partial x} & \frac{\partial P^0_{t-1}(w,w)}{\partial y} \end{pmatrix};\quad \Delta = \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} \tag{14}$$

The system of equations in (13) can be solved by $\Delta = (G^t G)^{-1} G^t D$.

When the feature pairs with subpixel accuracies become available, the motion parameters are computed using (8) and (10).

2.1.6 Motion Compensation

The result of the motion estimation process described in the last section is an estimate of the motion between two frames. The objective of motion compensation is to maintain a history of the motion estimates to create a stabilized sequence.

The manner in which motion estimation and compensation are performed defines the structure of the stabilization algorithm. We have compared two possible implementations: one that always uses two consecutive frames from the input image sequence to estimate the motion parameters, termed the frame-to-frame algorithm (FFA), and a second one that keeps a reference image and uses it to estimate the motion between the reference and the current input image, termed the frame-to-reference algorithm (FRA). We describe the first algorithm here.

2.1.7 The Frame-to-Frame Algorithm

The block diagram of the frame-to-frame algorithm is shown in Figure 2. The Laplacian $L[t]$ is built from the input camera image and sent to a delay buffer storage that keeps the previous pyramid $L[t-1]$ for the estimation process. The feature detection/tracking module

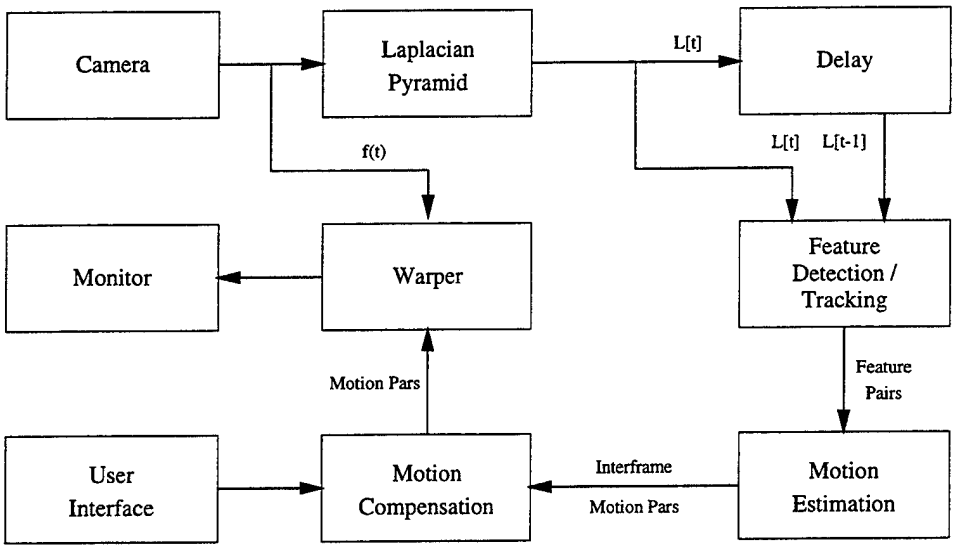


Figure 2: Block diagram of the frame-to-frame algorithm (FFA)

uses $L[t-1,0]$ to extract features that are used in hierarchical tracking. After the grid-to-grid matches are found, this module computes the subpixel-accuracy displacements that are used by the motion estimation module to compute the interframe motion. The motion compensation block must then compute the total motion of the camera by combining all the interframe estimates over time (since it started running, e.g., $t = 0$). The total motion is used to warp the current frame f_t, and the result of this operation is sent to the monitor. In this way, since all frames of the sequence are being warped back to $L[0,0]$, the resulting sequence is stabilized. To combine the motion parameters, assume that the transformation used to warp f_{t-1} is

$$\begin{pmatrix} X[0,0] \\ \vdots \\ Y[0,0] \end{pmatrix} = \mathcal{S}_0 \begin{pmatrix} \cos\Theta_0 & -\sin\Theta_0 \\ \sin\Theta_0 & \cos\Theta_0 \end{pmatrix} \begin{pmatrix} X[t-1,0] \\ Y[t-1,0] \end{pmatrix} + \begin{pmatrix} \Delta X_0 \\ \Delta Y_0 \end{pmatrix} \tag{15}$$

and the interframe motion is

$$\begin{pmatrix} X[t-1,0] \\ Y[t-1,0] \end{pmatrix} = \mathcal{S}_{t-1} \begin{pmatrix} \cos\Theta_{t-1} & -\sin\Theta_{t-1} \\ \sin\Theta_{t-1} & \cos\Theta_{t-1} \end{pmatrix} \begin{pmatrix} X[t,0] \\ Y[t,0] \end{pmatrix} + \begin{pmatrix} \Delta X_{t-1} \\ \Delta Y_{t-1} \end{pmatrix} \tag{16}$$

where $(X[t,i], Y[t,i])^t$ are the image coordinates of $L[t,i]$. Substituting (16) into (15), it is straightforward that the combined motion parameters $w_t = (\Delta X_t, \Delta Y_t, \Theta_t, \mathcal{S}_t)^t$ until frame

f_t will be

$$w_t = \begin{pmatrix} \Delta X_t \\ \Delta Y_t \\ \Theta_t \\ \mathcal{S}_t \end{pmatrix} = \begin{pmatrix} \mathcal{S}_0 \cos\Theta_0 \Delta X_{t-1} - \mathcal{S}_0 \sin\Theta_0 \Delta Y_{t-1} + \Delta X_0 \\ \mathcal{S}_0 \sin\Theta_0 \Delta X_{t-1} + \mathcal{S}_0 \cos\Theta_0 \Delta Y_{t-1} + \Delta Y_0 \\ \Theta_0 + \Theta_{t-1} \\ \mathcal{S}_0 \times \mathcal{S}_{t-1} \end{pmatrix} \tag{17}$$

Since features are tracked between consecutive frames, they do not need to be extracted in every frame. Figure 3 shows a simple scheme where the feature extraction algorithm is applied to every other frame instead of every frame. Suppose that the system uses frame f_t to extract features. This set of features can be used for tracking features backward between f_t and f_{t-1}. When f_{t+1} is acquired, the same set can be used for forward tracking between f_t and f_{t+1}. Finally, when f_{t+2} comes in, new features must be extracted and the process continues with backward tracking.

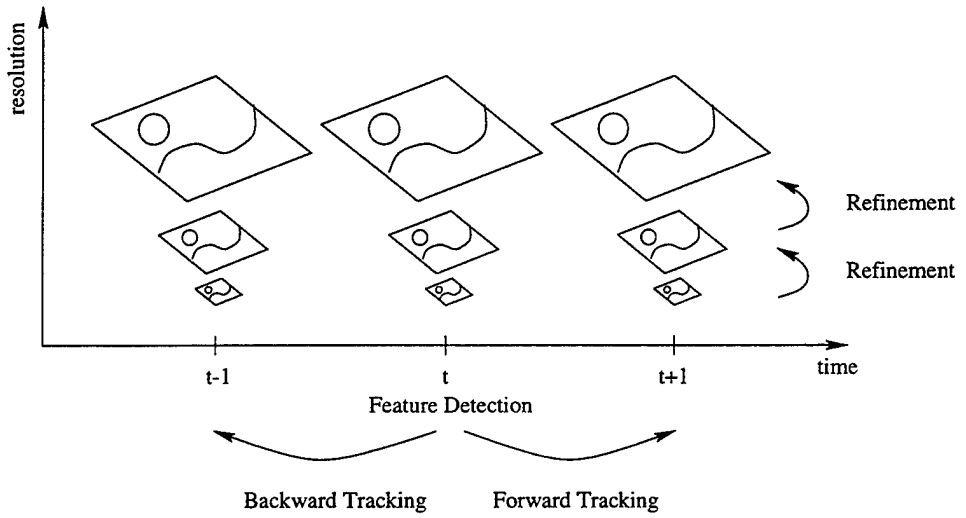


Figure 3: Feature detection and tracking

This strategy is very simple to implement and just requires a slightly more complex control structure. The forward tracking is done as described previously, but a few changes must be made for backward tracking. First, the same tracking algorithm is used by simply switching f_t with f_{t-1} and the motion based on the inverted pairs is computed. This process estimates the motion from f_t to f_{t-1}, but instead we need to compute the forward motion

from f_{t-1} to f_t. This can be done by

$$\begin{pmatrix} X_t \\ Y_t \end{pmatrix} = \mathcal{S}_t \begin{pmatrix} \cos\Theta_t & -\sin\Theta_t \\ \sin\Theta_t & \cos\Theta_t \end{pmatrix} \begin{pmatrix} X_{t-1} \\ Y_{t-1} \end{pmatrix} + \begin{pmatrix} \Delta X_t \\ \Delta Y_t \end{pmatrix}$$
$$\begin{pmatrix} X_{t-1} \\ Y_{t-1} \end{pmatrix} = \mathcal{S}_t^{-1} \begin{pmatrix} \cos\Theta_t & \sin\Theta_t \\ -\sin\Theta_t & \cos\Theta_t \end{pmatrix} \begin{pmatrix} X_t - \Delta X_t \\ Y_t - \Delta Y_t \end{pmatrix} \tag{18}$$

From (18) it is clear that the motion parameters from f_{t-1} to f_t can be written in terms of the backward motion parameters by

$$\begin{pmatrix} \Delta X_{t-1} \\ \Delta Y_{t-1} \\ \Theta_{t-1} \\ \mathcal{S}_{t-1} \end{pmatrix} = \begin{pmatrix} -(\cos\Theta_t \Delta X_t + \sin\Theta_t \Delta Y_t)/\mathcal{S}_t \\ -(-\sin\Theta_t \Delta X_t + \cos\Theta_t \Delta Y_t)/\mathcal{S}_t \\ -\Theta_t \\ 1/\mathcal{S}_t \end{pmatrix} \tag{19}$$

If the motion can be reliably estimated for large feature displacements, it is very simple to extend the backward/forward estimation scheme to utilize larger steps between feature extractions.

Due to the dynamic nature of the problem, the scene as a whole is under constant modification, so that after some relatively short time the original frame may not have anything in common with the current frame. An important characteristic of this algorithm is that it continues to compute the global motion even in these situations since, in general, the overlap between two consecutive frames is considerable. Of course the warping of the current frame using this global motion takes the image completely out of view. In such cases, the user interface module allows the user to reinitialize the system, so that another reference is taken. It is also possible to set motion limits, so that whenever the overlap between the reference and the current frame is not large enough, the system can reset itself automatically.

When the motion of the camera is predominantly lateral, i.e., the camera is moving perpendicular to its optical axis, the construction of a mosaic has some advantages [2]. It offers better visualization and also helps in the estimation process. Unfortunately, this technique does not help for forward (or backward) motion, i.e., motion along the optical axis, which is considerably more natural than lateral motion.

Besides reinitialization, the FFA can exhibit drifts due to the accumulation of error, which although very small between frames, can become considerable after long periods of time.

2.1.8 Hardware Description

The algorithm was implemented using a MaxVideo 200 board connected to a SUN SPARCstation 20/612 via a VME bus adaptor. The MV200 is a parallel pipeline image processing hardware system manufactured by Datacube Inc. The system is basically composed of the following dedicated devices:

1. *Architectural Adaptor Version B (AB device):* This provides a VME bus interface with the host computer, serving as the mother board for all the other devices. It allows data path programmability to and from all other MaxVideo processing resources through a 20 MHz crosspoint switch.

2. *Advanced Memory (AM device):* This provides basic image storage capabilities. One AB device can host up to 6 AM devices (Mem00 to Mem05), each having either 1, 4 or 16 MBytes of physical storage. Our board has 6 AM devices with 4 MBytes of physical memory each. These devices are typically used as buffers for several image processing operations, including image acquisition and display.

3. *Analog Generator (AG device):* This provides display and graphics overlay capabilities. It can be programmed so that it can be connected to many different types of monitors.

4. *Advanced Pipeline Processor (AP device):* This processor can perform three basic statistical operations (histograming, feature listing, and modified Hough transform), morphological operations, and neighborhood multiply and accumulate (NMAC) operations that can perform convolutions of an entire image with an (up to) 8×8 convolution mask. Our AP device also hosts a second-order polynomial warper module that is able to perform second-order polynomial transformations in real time.

5. *Analog Scanner (AS device):* This device digitizes the analog input video signal, storing the image in its own physical storage element.

6. *Arithmetic Unit (AU device):* This device is able to perform several linear, non-linear, and statistics processing operations.

Each device consists of a variety of processing elements, including input/output ports, memory elements, multiplexers, look-up tables, etc. The MV200 image processing system comes with ImageFlow, a collection of C routines that controls and connects these processing elements. The task of the programmer is to set the control parameters and connect all the necessary elements in order to perform the desired computations on the input image stream.

The connection of elements is done within the elements of one device. Connections between devices are made through the crosspoint switch of the AB device. The basic processing unit, the *pipe*, that defines a path between two memory storage elements (typically from the AM device). To define a pipe, all elements within the pipe must be connected, including the connections between devices, and the control parameters of each element must be appropriately set. After the pipe is ready, it can be fired (once or continuously), so that the data from one memory element flows through the pipe to the second memory element, where the processed data can be accessed and further processed.

For example, a simple convolution can be defined by three pipes: One pipe connecting the AS device (which is connected to a camera) to a memory storage device, say Mem05; a second pipe for the convolution, connecting Mem05 to Mem00 through the convolution element in the AP device; and a third pipe for display purposes, connecting the result of the convolution stored in Mem00 to the AG device (which is connected to a monitor).

2.1.9 Implementation of the Frame-to-Frame Algorithm

Figure 4 shows a block diagram of the Datacube implementation of the frame-to-frame algorithm. The simplicity of the system contributes to its fast performance. We initially built all the modules in C++ for simulation purposes and then ported the appropriate modules to the Datacube. To optimize performance, communication between the Datacube and the host computer (the SUN workstation) was kept to a minimum.

The acquisition pipe stores frames of size 512×480 continuously into Mem05; they are digitized by the AS device. A one-shot pipe links Mem05 to Mem04 through the AP

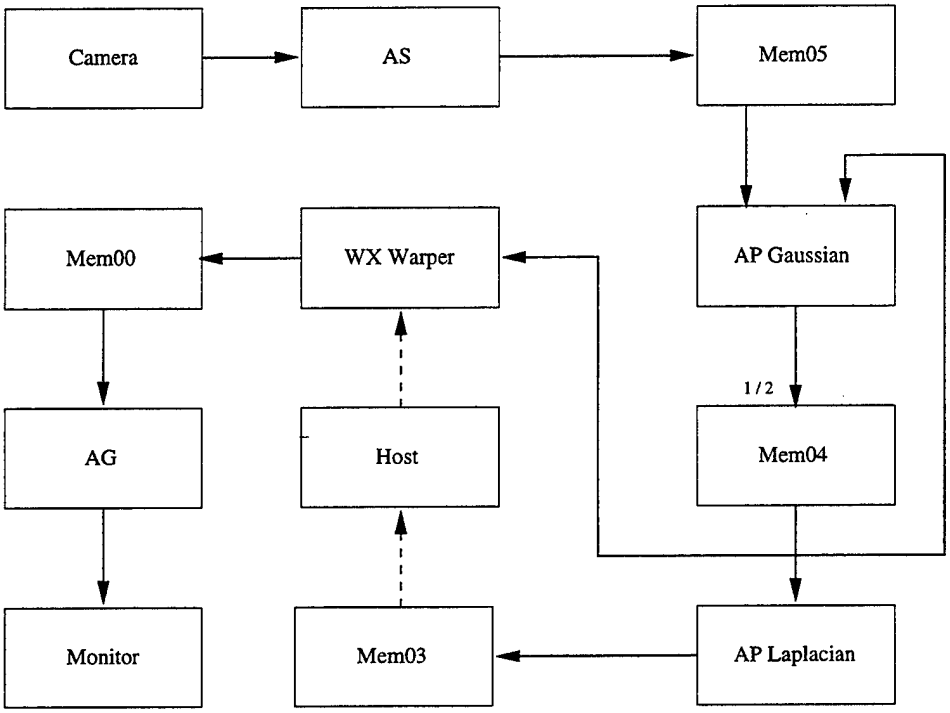


Figure 4: Block diagram of the Datacube implementation of the FFA

convolution element, loaded with a 5×5 Gaussian convolution kernel. The input port of Mem04 is set to subsample the input by a factor of two. To continue the subsampling process, a looping pipe around Mem04 creates several subsampled Gaussian images, until a 32×30 image size is reached. To avoid superposition with finer resolution images, the coarser images must be carefully placed into Mem04. All images in Mem04 are then convolved with a 5×5 Laplacian kernel and stored in Mem03. The 32×30, 64×60 and 128×120 images are sent to the host computer. Features, typically five, are extracted from the 128×120 image, and tracked within subpixel accuracy. We allow for local displacements of ± 3 on each level, so that the system is able to support a maximum displacement of ± 21 pixels in the 128×120 image. After the computation of the motion parameters, they are used to set the warper module in the AP device. The 128×120 image stored in Mem04 is warped and sent to Mem00. Finally the result is displayed by the pipe from Mem00 to the AG device. If the parameters are appropriately scaled, the warper can be easily set to warp any image, including the original one stored in Mem05. The host also controls the forward/backward scheme and the user interface.

2.1.10 Experimental Results

This section presents experimental results obtained from a video sequence taken from the NIST HMMWV. Unfortunately, still images are not the most appropriate way to display the results of such a dynamic process. Those readers with access to www browsers may want to look at the MPEG movies, containing a few original and stabilized sequence samples, that are available at:

http://www.cfar.umd.edu/~carlos/stabilization.html.

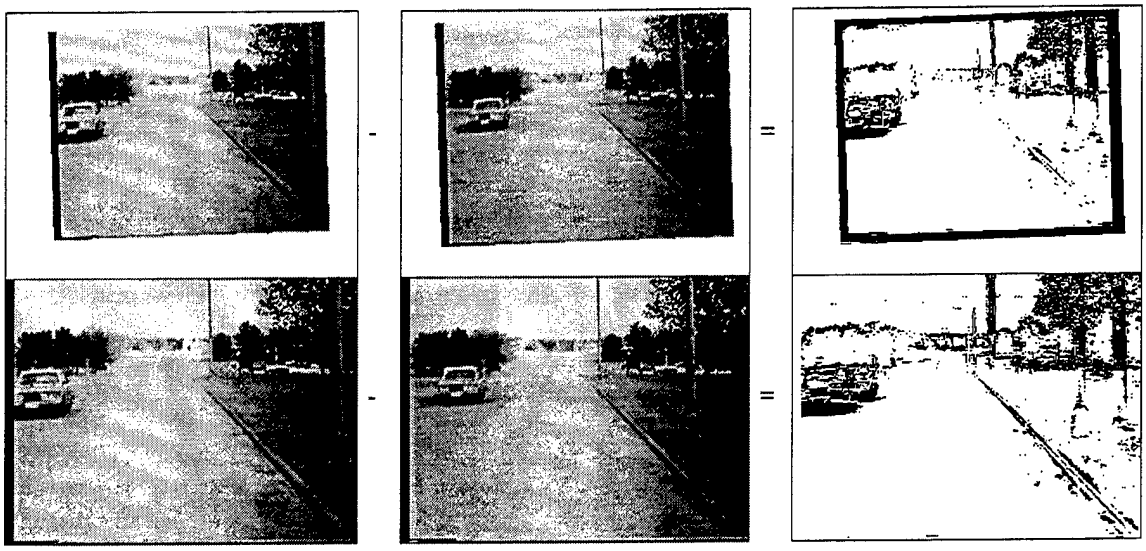


Figure 5: Stabilization results.

Since our stabilization technique is supposed to stabilize the entire reference image, the difference between two stabilized frames should be close to zero everywhere; this is not true in the case of the original sequence due to the motion of the camera, which causes misalignments in the images. The top of Figure 5 shows two stabilized frames and their differences, while the bottom shows the corresponding frames from the original sequence. For illustration purposes, the differences were thresholded, so that the black spots correspond to high-intensity discrepancies. In this case, the number of black spots in the difference of images can be regarded as an error measure that stabilization tries to minimize.

The original sequence was recorded from the NIST HMMWV while moving forward. That is the reason for the shrinking in the stabilized frames. It can be seen that objects closer to the camera (e.g., a car to the left and a lamppost to the right) are quite noticeable

(the error is large) in the stabilized sequence because their motion does not fit the 2D model, but it is clear that the region around the horizon practically disappears (the error is small) in the difference image. Observe that since we are tracking distant features, the system is quite robust to perspective distortions that occur in other parts of the image.

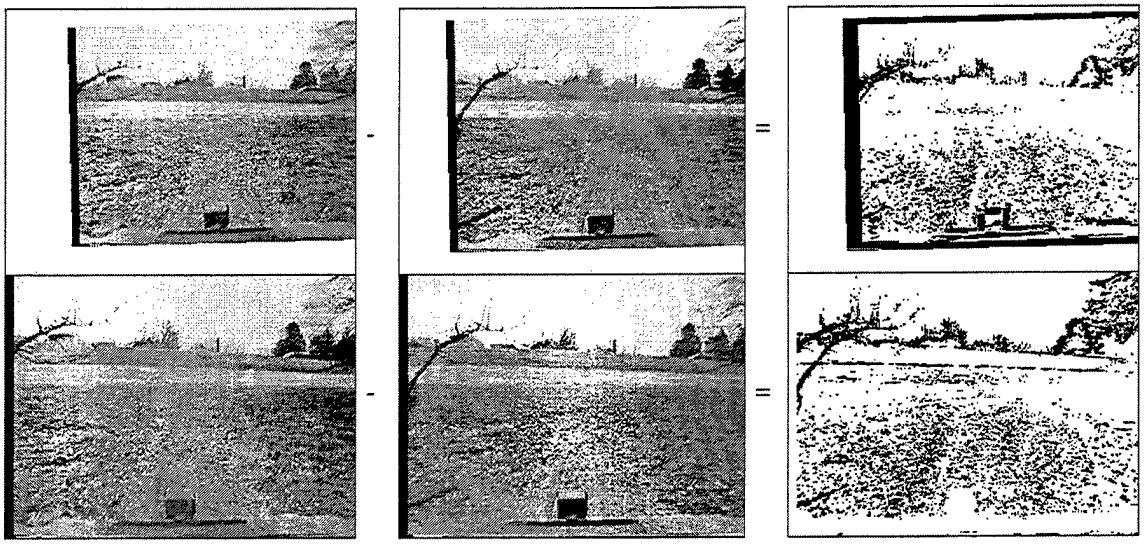


Figure 6: Stabilization results for off-road navigation.

Figure 6 shows another part of the tape where the vehicle is navigating on rough terrain. Again, the error is much smaller around the regions close to the horizon, but high on the bottom regions due to the different image flow of the grass pattern, which is not appropriately compensated by the 2D model. The results are apparently good enough for visualization, but segmenting independently moving objects directly from the difference of stabilized frames is not straightforward, mainly due to regions that do not fit the 2D motion assumption, since they are also segmented after the difference image is thresholded. Temporal median filters combined with velocity-tuned filters are used in [9] to detect independently moving objects (IMOs). Qualitative approaches [10, 11] to the detection of IMOs could also benefit from electronic image stabilization.

During these tests the system was set to use search windows and SSD windows of the same size (7×7). For these settings, the system is able to process about 20 frames per second. Table 1 shows how the frame rate degrades when the search window is increased and the SSD windows are kept constant. Our host computer is running SUN-OS 4.1, which is

not the most appropriate operating system for real-time applications. Thus, for each search window, three trials of at least 1000 frames were processed and the average result is shown.

Table 1: Performance evaluation of search window sizes.

Local Search [pixels]	Global Search [pixels]	Frame Rate [frames/s]	Image Velocity [pixels/sec]
±1	± 7	21.8	152.6
±2	± 14	20.8	291.2
±3	± 21	19.7	413.7
±4	± 28	18.4	515.2
±5	± 35	17.0	595.0
Hansen *et al.*	± 32	10	320

It is possible to consider the frame rate alone as a basis for comparison, but frame rate and robustness to large pixel displacements can be combined by considering the maximum pixel velocities supported by the system. The rightmost column in Table 1 gives the maximum pixel velocities that each setting is able to process. This might suggest that the system is more robust when set to search displacements of magnitude 5 instead of 3, because despite the loss in frame rate there is a considerable gain in feature velocity that the system is able to track. Unfortunately, when the search space is increased, the narrow SSD window being used is more likely to find a false match. Increasing the SSD window provides better discrimination between features and decreases the probability of false matches. On the other hand, the frame rate drops considerably.

The last row of Table 1 shows the performance of the stabilization system presented by Hansen *et al.* [5]. Their system is able to stabilize images with velocities of 320 pixels per second, running at 10 frames per second. Despite the better performance figures of our system for a search window of size ±3, we expect that their system may be more precise in the computation of the motion parameters since several local patches contribute to the refinement and estimation of the parameters, while we use a very limited set of feature points that are simply tracked and used for estimation. On the other hand, our technique can handle sequences with regions that do not fit the 2D model if the features are constrained to distant points of the scene. Also, for real-time applications, faster frame rates (closer to

video frame rate) might be more appropriate.

The fast frame rate was obtained in part by reducing the number of features to be tracked (typically set to 5). This explains why the system is so sensitive to false matches and bad features (features in regions that do not fit the 2D model). Assuming that translation is dominant, the flow of distant (or vanishing) points should be very small. So, to reduce the sensitivity of the system to bad features and false matches, the motion parameters are computed based on the three features with minimal displacements.

2.1.11 Detection of Independently Moving Objects

Image stabilization renders the background of the image approximately stationary. In order to overcome the effects of residual motions, we implemented a *temporal median filter.* This filter creates a *median image* that is composed of the median values of the last k frames. In a sequence of pixel gray levels from k frames, the gray levels arising from an independently moving object tend to be outliers with respect to the median of the sequence, so that they at least partially disappear from the median image. A simple image differencing scheme can then be applied to detect independent motion on a frame-by-frame basis.

A *filtered image* is a binary image obtained by thresholding the difference between the median image and the current frame. This process tends to erase the background and highlight the locations of independently moving objects. These filtered images still contain noise (due to imperfect stabilization) and spots corresponding to close scene objects that appear to move due to motion parallax. Velocity-tuned filters are used to reject the stabilization noise; motion parallax spots are also rejected if their apparent motion in the image is out of the velocity range for which the filter is tuned.

Assume that a filtered image f_i contains several spots, and we want to select only those that move linearly at a rate of p pixels per processed frame. If such spots also appeared in the previous frame, f_{i-1}, by the time frame f_i is captured these spots must have moved p pixels away from their f_{i-1} positions and are therefore located in frame f_i somewhere on *circles*, p pixels in radius, centered at their f_{i-1} positions. If we select the pixels that correspond to the intersections between the spots of f_i and the spots generated by replacing the spots of

f_{i-1} by the appropriate circles (predicted positions), we obtain good candidates for regions moving at p pixels per frame. This scheme can be extended to include more than two frames, since spots in frame f_{i-2} should be $2p$ pixels away, and spots in frame f_{i-j} should be jp pixels away. Implementation issues regarding these filters are given in [9].

2.1.12 Experimental Results

Figure 7 shows a frame of a video sequence taken from a moving vehicle and Figure 8 shows the thresholded difference between the four-frame temporal median and the stabilized instance of that frame. Finally, Figure 9 shows the stabilized frame superimposed on the output of the velocity-tuned filters integrated over four frames.

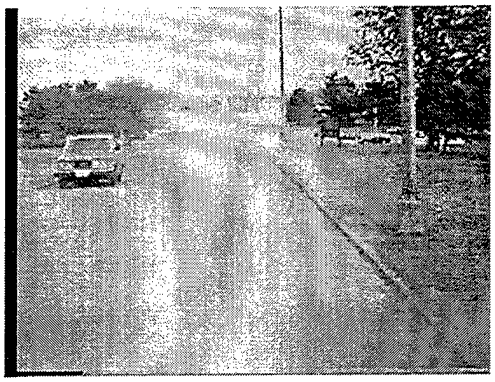


Figure 7: Frame from video sequence.

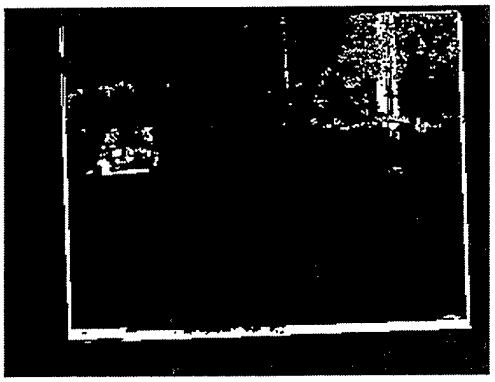


Figure 8: Thresholded difference between temporal median and stabilized frame.

Figure 9: Stabilized frame with superimposed moving object regions.

2.2 University of Rochester

(Randal Nelson, Rajesh P.N. Rao)

Work at the University of Rochester addressed the RSTA goal of detecting and tracking independently moving objects from a moving platform. The detection of independently moving objects is a critical task for RSTA subsystems. Because objects that move independently represent possible threats, it is important to flag them as rapidly as possible and then track them so that identification systems can be brought to bear on them. We have been engaged in a project whose goal is to design, implement, and test a general framework for utilizing visual motion for the detection and recognition of events and objects. Overall, we have been developing a three-step process for motion recognition that includes detection, tracking, and recognition phases. Of these steps, the detection and tracking components are of most immediate interest to the RSTA demos, and we have been engaged in porting previously developed algorithms for these aspects of the problem to hardware on the NIST vehicle, and evaluating the algorithms under various field conditions. The hardware and mechanics on the NIST vehicle are consistent with those of the current RSTA vehicles, and thus provide a valid testbed relative to RSTA demo goals.

The fast detection of potentially significant motion events is based on identifying violations of qualitative rigid-world constraints. This provides a uniformly applicable strategy by which small regions of the scene can be selected for more thorough inspection. In previous work, we produced real-time algorithms for detecting independently moving objects from a moving platform [10]. These techniques are more general than those based on affine

stabilization of the visual field, and can function in situations containing substantial motion parallax at different depths, and skewed, non-planar radial flow (such as that produced in the near field during locomotion through hilly terrain), which cause problems for the affine methods. They can thus serve to augment affine stabilization algorithms, which have previously demonstrated their value in regimes where they are valid. We have ported these algorithms to a platform consistent with the hardware on the RSTA vehicles, and have evaluated their performance, both in isolation and in combination with low-level stabilization algorithms developed at Maryland.

The tracking step involves stabilization of the area of interest through active visual processes such as fixation and tracking to place the motion of interest in a canonical form that facilitates the final recognition procedure. The techniques of most immediate interest for RSTA involve the tracking of independently moving rigid objects. We have developed real-time algorithms for instantiating and maintaining hypotheses about the positions, extents, and motions of such objects on the basis of the output from the independent motion detection system. We have also developed techniques for accomplishing this for objects that move in a complex manner, such as people or animals [13].

The identification step locates regions of interest via a more detailed analysis of motion. We have developed techniques based on *temporal texture analysis*, where we extract statistical spatial and temporal features from approximations to the motion field and use techniques analogous to those developed for gray-scale texture analysis to classify regional activities. Some results in this area are described in [11]. In a second approach, which we term *activity recognition*, we use the spatial and temporal arrangement of motion features in conjunction with simple geometric image analysis to identify complexly moving objects such as machinery and locomoting people and animals [14]. The remainder of this section concentrates on the motion detection processes.

Detection of moving objects is of critical importance to biological and robotic systems, both because such objects are frequently of primary interest to the system, and because dealing with them involves hard real-time constraints—the world won't wait while you think. A method of detecting independent motion, or motion having certain other qualitative characteristics such as periodicity, is thus valuable as a method for directing more sophisticated

(and costly) processing to areas where it can be most effectively utilized.

In previous work, we developed methods for detecting three qualitative types of motion. The first technique, which we term **constraint ray filtering**, provides a robust method of detecting independently moving objects from a moving platform when information is available about the platform motion [10]. The method is based on the observation that the projected motion at any point on the image sphere is constrained to lie on a half line (ray) in local velocity space whose parameters depend only on the observer's motion and the location of the image point. The second method, termed *animate motion detection*, allows rapid detection of animate objects with no information about the movement of the platform [10]. It is based on the observation that animate moving objects typically *maneuver*, that is, they or their component parts follow trajectories for which the projected velocity changes rapidly compared to the velocity change due to self-motion. The third method allows detection and tracking of objects whose motion has a periodic component, such as walking or running animals, oscillating machinery, etc. [13]. It is based on a Fourier transform technique.

These techniques can be used to isolate motion for identification by later recognition processes. The responses of the different qualitative detectors yield an indication of the sort of recognition process that should be assigned to the movement of interest. For example, detection of local, highly periodic movement would suggest the use of a phase-based structural classifier, while a distributed, non-periodic motion would suggest the use of temporal texture techniques.

The first two techniques were originally implemented as real-time systems on Datacube series 10 hardware, and demonstrated in a laboratory setting. Of these, the first, constraint ray filtering, is of the most immediate interest to the RSTA goal of detecting moving vehicles. We have ported this algorithm to hardware on the NIST vehicle, which is compatible with the hardware on the Martin Denver demo vehicles, and initiated the evaluation of the algorithm using outdoor driving sequences acquired from both the NIST vehicle and other vehicles.

Representative results of the moving object detection algorithm are illustrated in Figures 10 a–b. Figure 10a shows a frame in a video sequence acquired from a forward moving mobile platform. The independently moving objects are the cars moving left to right near the horizon. Figure 10b shows the superposition of the pixels detected by the independent

motion detector onto the video frame. These pixels lie on the car moving across the image.

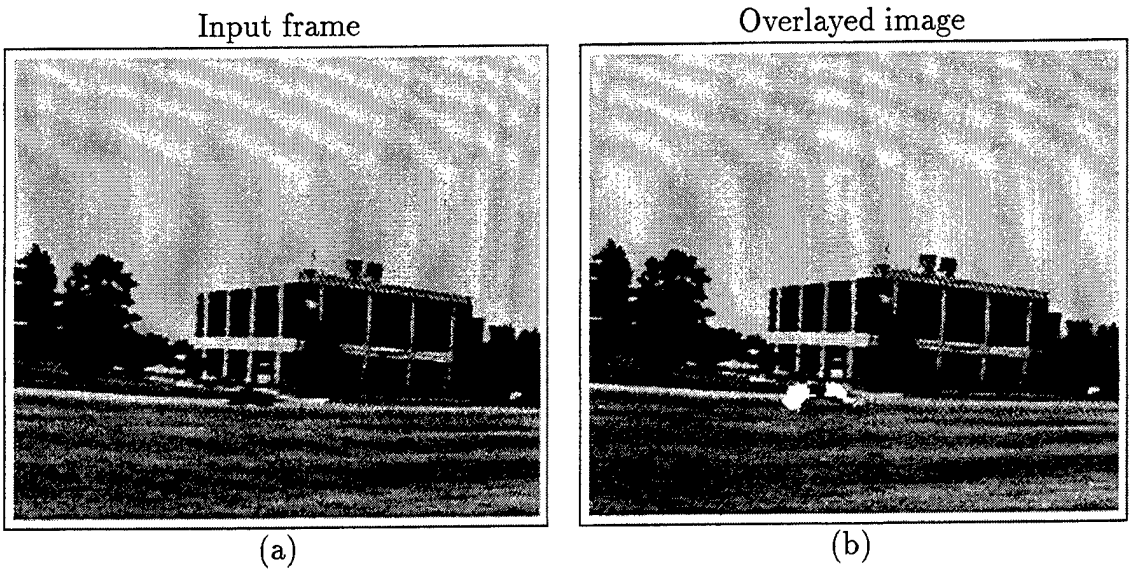


Figure 10: (a) Original image. (b) Independently moving pixels overlaid on (a).

Theoretical analysis of the algorithm indicates that, although the underlying technique is capable of detecting independent motion in an arbitrary environment under arbitrary motion, the instantiation of the algorithm on the Maxvideo hardware has certain limitations. In particular, the choice of a first-order gradient-based flow estimation technique is dictated by the operations that the Maxvideo performs efficiently (namely, convolution), and this limits the accuracy of the motion field estimation. Further analysis indicates that when the magnitude of the motion due to vehicle movement exceeds the magnitude of the independent motion by more than a factor of two, detection is unreliable. We can recognize this situation and avoid false positives, but genuine independent motion may then go undetected.

The theoretical performance is borne out by field tests. When driving on roads, or slowly on relatively smooth terrain, the algorithm detected other moving vehicles in a variety of situations. However, at high velocity, and over rough off-road terrain, the detection limit is frequently exceeded, and independently moving objects are missed. Further analysis revealed that this effect is primarily due to rapidly changing vehicle pitch, with smaller effects from roll and yaw.

Two approaches can be used to resolve this situation. The first is to use a more accurate

motion estimation algorithm. However, this is probably not practical in real time using the existing hardware, though we have explored the use of a multi-resolution gradient-based algorithm. (Advances in hardware may change this picture for the next generation of vehicles, but for the moment we are limited to the current Maxvideo system). The second approach notes that the dominant source of large motions that swamp the detection algorithm is vehicle rotation, which is removable by stabilization techniques of a sort that have already been demonstrated.

We have instantiated the second approach, using several stabilization algorithms developed at the University of Maryland as pre-processors to the motion detection system. The different algorithms were compared, and the best of them selected for our demonstration. We have also developed a predictive tracker that will use the (pixel map) output of the independent motion detection system to circumscribe and track potential target objects. These regions of interest will ultimately serve as inputs to the next phase of the system where recognition and higher-level planning are performed.

2.3 University of Pennsylvania

(Ruzena Bajcsy, Ulf Cahn von Seelen)

The University of Pennsylvania's research was concerned with camera control for tracking acquired targets. The controlled axes include mechanical degrees of freedom (pan, tilt) as well as an optical degree (zoom). The hardware platform consists of a TRC BiSight binocular camera platform controlled by a PMAC-VME motion controller that is connected to a Sun workstation via shared memory.

While object tracking by panning and tilting a camera is well known, the use of zoom in tracking is largely unexplored. For RSTA on the Move we want to maximize the spatial resolution of the tracked object while maintaining acquisition. This involves optimizing the tradeoff between spatial resolution and tracking performance. The closer the camera zooms in on the target, the faster the target moves in the image, and the harder it becomes to maintain acquisition of it.

To our knowledge, there exist only a few publications that deal with the control of zoom

for tracking. In [7] the zoom is used to achieve a desired object image size. A fuzzy controller combines estimates of the diagonal extent of the object, the variance of the object velocity, and the confidence of the shape estimate to compute a suitable focal length. The influence of the velocity variance ensures that the camera does not zoom in too closely if an object's motion varies greatly, in order to safely maintain acquisition.

In [6] the authors use a robot arm and camera zoom to achieve a desired image feature configuration. The focus of the work is on integrating the zoom into the control as a redundant mechanical degree of freedom, as the authors assume that the image Jacobian and thus the 3D positions of the image features in the world are known. This assumption abstracts from the main problem of using zoom in tracking, namely finding an image-based measure on which to servo the focal length.

In the *PennEyes* system [8] we have used various image-based measures to maintain the apparent size of a target in an image. In the current version we use cross-correlation to identify the target. This approach is more general, but it does not provide a ready estimate of the apparent target size. We work with the object distance instead, which we estimate by triangulation from the two camera views. Using our calibration of the zoom lens, we can compute a new focal length when the target distance changes so that the image size of the target remains constant. Figure 11 shows a typical run of the system in which the focal length is increased so that it compensates for the target motion away from the camera head.

With the expertise gained from zooming for size constancy, we can approach the problem of zooming for scale change. Changing scale poses increased demands on target identification and localization because commonly used approaches such as cross-correlation are not scale-invariant. Alternatives include feature-based tracking (e.g. [17]) or the use of adaptive correlation templates (e.g. [12]).

2.4 National Institute of Standards and Technology

(Martin Herman, David Coombs, Sandor Szabo, Tsai-Hong Hong)

NIST was responsible for developing the vision processing platform, assisting in integrating University software onto the platform, and running the platform on vehicles at the NIST

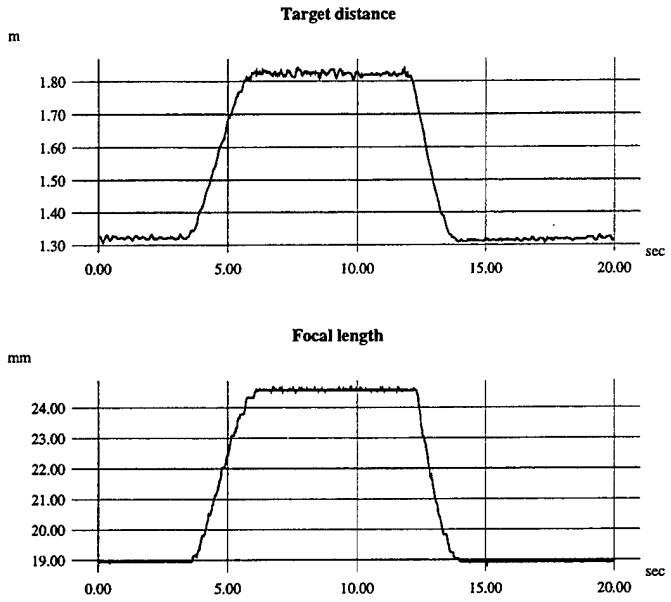


Figure 11: Focal length responding to changing target distance.

facility. In addition, NIST collected video data using the NIST HMMWV, and worked on target tracking and gaze control software. NIST completed development of the platform and worked with the Universities to demonstrate two components of RSTA on the Move: image stabilization and independent motion detection.

The NIST vision processing platform is based on industry-standard components which allow us to integrate, test and distribute results with minimal amounts of effort. For example, approximately one hour was required to initially install and run software from each University. This allows us to spend a considerably greater portion of our time in analyzing and improving system performance. The platform, designed for mobile applications, was easily shipped, set up and demonstrated at ARPA's UGV Demo C in Denver in July 1995. Since then we have completed the power conversion of the system so that we can now run from vehicle DC power sources as well as conventional AC sources. We are close to upgrading our computing system to three processors and to Solaris 2.4 which will allow us to take full advantage of symmetric multiprocessing and real-time scheduling. Our design, integrating the RSTA software components and taking full advantage of multiple general purpose processors, and multiple specialized image processors is almost complete. By early Spring of 1996 we were able to perform experiments on the NIST HMMWV on a regular basis.

Because of our close involvement with the ARPA Demo II program and the Army Research Laboratory, we feel that the results of our effort can be readily integrated into future DoD mobile robot applications.

The NIST vision processing system, designed for RSTA on the Move applications, is capable of performing sophisticated experiments in mobile vehicle applications. The system consists of a TRC UniSight/BiSight camera head (pan/tilt/vergence), a Datacube MV200 image processor for low level image processing (acquisition, filtering, overlays, etc.), and a suite of fast SPARC processors (for motion analysis, tracking, etc.). Both the image stabilization algorithm from the University of Maryland and the independent motion detection algorithm from the University of Rochester rely on image processing taking place in both the specialized Datacube environment and the general purpose SPARC environment. The University of Pennsylvania tracking software relies on the specialized motion controller for the camera head (a Delta Tau PMAC motion controller board) and the SPARC environment.

The system components of the vision processing platform consist of a VME-based card cage housing all the processor boards, a two gigabyte ruggedized hard disk, and electronics for the cameras and the TRC camera head. The VME provides power and fast communications between a Themis SPARC 10MP processor board, the Datacube MV200 image processor board, and the Delta Tau PMAC motion controller board.

The Themis board is outfitted with an 80 MHz and two 90 MHz HyperSparc processors. We dedicate one processor to image stabilization, one to independent motion detection, and one to tracking. The processors are run in pipeline mode with the results from one stage being fed to the next stage. Additional parallelization within stages could be added to reduce the overall latency. By changing from Solaris 1.1 to Solaris 2.4, we can take advantage of the multithreading libraries and the real-time scheduling facilities. Changing to a complete Solaris system also allows us to migrate away from having separate operating systems for development and real-time applications, thus further simplifying integration. All of the code is written in C/C++ and makes use of the GNU Free Software Foundation environment. We have installed and tested software from the University of Rochester and the University of Maryland under this environment without any problems.

The Datacube MV200 is practically the industry de-facto standard for real-time vision

processing. We have installed a complete programming environment for the MV200: Imageflow, Advanced Imaging Tools, WitFlow, and Veil. We have also installed a miniwarper. Both the University of Rochester and the University of Maryland algorithms require the MV200, but we can pipeline the algorithms using two MV200 boards.

Table 2: Detection results of three stabilization algorithms.

Algorithm	Threshold	% targets detected	# frames to acq. target	Avg. false alarms/frame	% targets segmented
Projection	17	0	NA	NA	NA
FTA 1	12	0	NA	NA	NA
FTA 2	12	100	7	1	67

The Delta Tau PMAC motion controller board is used for control of the TRC head. NIST has experience in this area, having built a head (TRICLOPS) in the past, and we can incorporate previously developed head control software into the RSTA application. We have received software from the University of Pennsylvania for controlling the head at a low level and have also developed our own software for computing quintic-based smooth trajectories.

The NIST system is completely self-contained. Each of the components is designed for modularity, having its own dedicated power conditioner to run off DC power sources. Sufficient power exists to run a fully configured four-processor SPARC 10MP, three MV200's, the motion controller board, and an additional I/O processor designed for a potential vestibular sensor system. All the components are housed in a sealed, ruggedized enclosure designed for outdoor vehicles. NIST, with the support of the Army Research Laboratory, also maintains a fully robotic HMMWV which enables us to perform experiments on a regular basis.

Data Collection. NIST has collected over six hours of videotape from color CCD cameras rigidly mounted on the HMMWV, driving at up to 40 kph on- and off-road at the NIST site in Gaithersburg, MD between December 1993 and June 1995. The terrain includes campus roads, fields and woods. Civilian vehicles can be seen driving on the NIST grounds and on the surrounding roads (including highway I-270) at ranges up to 2000 m. Pedestrians and deer are also visible on occasion. The cameras are rigidly mounted on the vehicle in forward-looking and oblique-looking (60 degrees off heading) orientations. The

lens focal lengths used range from 5 mm to 75 mm. No stabilization was used (neither mechanical stabilization of the cameras nor digital stabilization of the video) and image jitter is particularly noticeable with longer focal-length lenses.

3 Supporting Basic Research

In addition to the integration activities outlined in the previous section, each of the consortium members also pursued a program of basic research on enabling technologies for RSTA on the Move. In this section we provide brief descriptions of some of these research projects.

3.1 Performance Characterization of Image Stabilization Algorithms—University of Maryland

We have carried out a comparative study of image stabilization algorithms in the context of an automatic target tracking system. This study was conducted jointly with the Army Research Laboratory (ARL). The goal is to perform target acquisition through a process of background suppression and motion estimation. In order to accomplish this it is important that the input sequence be stabilized so that image motion due to camera motion as the camera is panned, or as the camera moves through the scene, is compensated for.

Three stabilization algorithms were compared with respect to target false alarm and false dismissal rates, time to acquisition of targets, and a gross measure of the accuracy of target segmentation.

- The first algorithm was developed at ARL to compensate for wind loading on an unmanned robotic platform. It is a simple algorithm that can only estimate integer image translations, and operates on normalized row and column projections of consecutive frames in the video sequence.
- The second algorithm was developed at the University of Maryland; it is a multiresolution version of the algorithm described in Section 2 of this report.
- The third algorithm was a generalization of the second one; it uses longer image sequences for motion estimation.

In spite of the fact that the stabilized image sequences obtained from the three algorithms were perceptually almost indistinguishable, there were dramatic differences in performance between the first two algorithms and the third. The first two algorithms had unacceptably high false dismissal and false alarm rates on the tested image sequences. On the other hand, when the target tracker was integrated with the third algorithm, it achieved a 0% false dismissal rate and a 1% false alarm rate on real IR sequences. Details of this study are reported in [1].

Figure 12 shows a typical image from one of the real FLIR sequences employed in the experiments. The target, near the top of the image, is outlined in a box. Table 2 compares the detection results of the three stabilization algorithms on one of the FLIR sequences. Again, even though there is little perceptual difference between the stabilized sequences produced by the three algorithms, the impact of the small differences on target acquisition and false alarm detection rates were quite significant.

Figure 12: Typical image from a real FLIR sequence.

3.2 3D Model-Based Image Stabilization—University of Maryland

We have studied the use of combined visual cues and dynamic models for the stabilization of calibrated or uncalibrated image sequences [19].

Parameters relevant to image warping are estimated by combining information from different tracked tokens, namely points and horizon lines. These parameters are simply the

camera rotational velocity if intrinsic camera parameters are available, or the projectivity coefficients, in the uncalibrated case. Image plane displacements of distant feature points may unambiguously characterize rotational motion. However, such points are sometimes difficult to detect and track, due to the absence of sufficient intensity gradient information. Horizon lines, when present, on the other hand, constitute very strong visual cues, requiring relatively simple operations for their tracking. These tokens are therefore both used in our stabilization scheme.

We have investigated how to use temporal information in a sequence to facilitate the estimation of parameters of interest. Image stabilization is a process closely related but not equivalent to image registration. Registration techniques can be extended for stabilization purposes. Image stabilization is inherently different in that it allows the use of dynamical information over long temporal windows. In unmanned ground vehicle application, cameras are mounted rigidly on the platform. The rotation of the vehicle arises from the rotational movement of the vehicle. It is therefore possible to employ a kinetic law which captures the rotation of the platform to model the temporal behavior of the parameters of interest. However, with the aid of visual cues, simple kinematic laws become feasible. We therefore use a kinematic law to model the temporal behavior of relevant parameters.

Specifically, for calibrated sequences, since the intrinsic parameters of the camera are known, the perspective projection model which describes the relationship between 3D scenes and their 2D projections can be used to characterize the projection of both distant points and horizon lines. After the points and horizon lines are tracked over the sequence, they can be used along with a kinematic law to estimate the rotational parameters. Based on the estimated parameters, a stabilized sequence is generated.

For uncalibrated sequences, to integrate distant points and horizon lines, a different description of the movement of horizon lines is employed. This leads to the estimation of eight projective coefficients, in order to stabilize the uncalibrated sequence. However, the estimates of these projective coefficients are very sensitive to the tracking of points and lines. On the other hand, the intrinsic parameters are often approximately known. Instead of estimating the eight projective coefficients, our uncalibrated stabilization scheme is then similar to the calibrated scheme and concentrates on estimating the three rotational

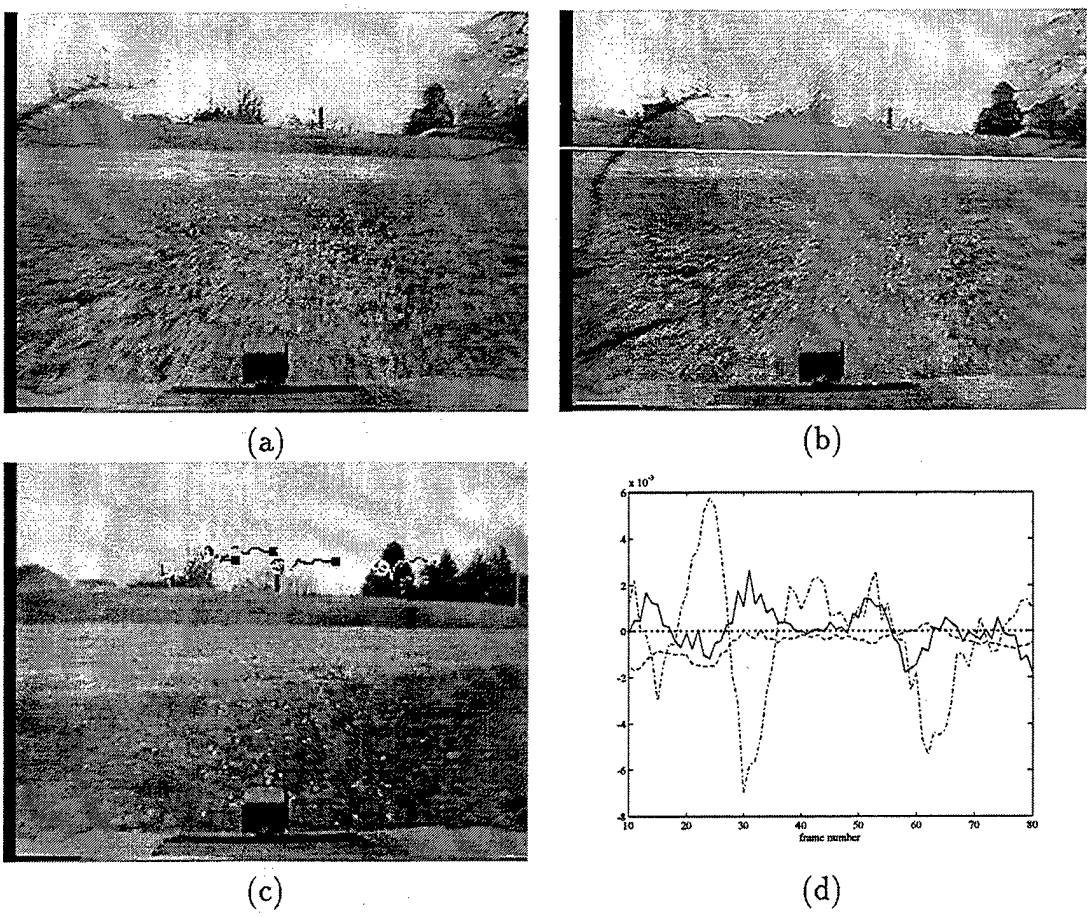


Figure 13: (a) A sample image from a sequence, (b) an image with horizon line detected, (c) an image with point trajectories, and (d) a plot showing the estimated 3D rotational parameters

parameters while assuming the approximate intrinsic parameters.

Both schemes have been tested on real sequences with good results. The results of this research are illustrated in Figures 13(a–d). Figure 13a shows a sample image from an outdoor sequence. Figure 13b is the same image with the horizon line superimposed. Figure 13c shows the point trajectories for features in that sequence and Figure 13d is a plot of the estimated 3D rotational parameters.

3.3 Perception of the UGV's Environment—University of Maryland

Our work on RSTA on the Move has concentrated on the interplay between the recovery of three-dimensional motion information and the recovery of descriptions of the immediate

environment of the UGV. Regarding the perception of 3D motion we have implemented a set of techniques that recognizes a collection of global patterns of robust spatiotemporal measurements. The localization of these patterns, which are independent of the structure of the scene in view, encodes the underlying 3D motion parameters, enabling stabilization through the interpolation of the UGV's intended motion in the temporal evolution of the measured motion. Figure 14 shows experiments using the approach developed in [4] with real data collected from the vehicle.

A recent technical development related to the perception of the UGV's environment is the concept of iso-distortion surfaces, a framework for studying the relationship between the computation of 3D motion and depth from a sequence of images [3]. The underlying conceptual theme is that motion errors (e.g., errors between retinal motion and perceived 3D motion) affect depth estimates systematically. The understanding of the geometry of this distortion of depth is essential for understanding the interplay between 3D motion and shape processing and thus for interpreting visual motion. The introduced framework characterizes this relationship via a family of iso-distortion contours, which describes the loci over which depths are distorted by the same amounts. Figure 15 shows an example of the iso-distortion contours that result from intersecting the iso-distortion surfaces with the Zx-plane.

The tool of iso-distortion surfaces allows us to study the very practical problem of calculating the precision of an inertial system that is sufficient for obtaining unbiased estimates of the vehicle's heading direction, using algorithms that combine inertial and visual measurements. The process is explained in Figure 16.

3.4 Fast, Filter-Based, Object Location and Identification— University of Rochester

We have developed a visual location and identification system [16] based on efficiently computable iconic representations. The system uses two primary visual routines, one for identifying the visual image near the fovea (*object identification*), and another for locating a stored prototype on the retina (*object location*). The iconic representations are based on high-dimensional feature vectors obtained from the responses of an ensemble of *steerable*

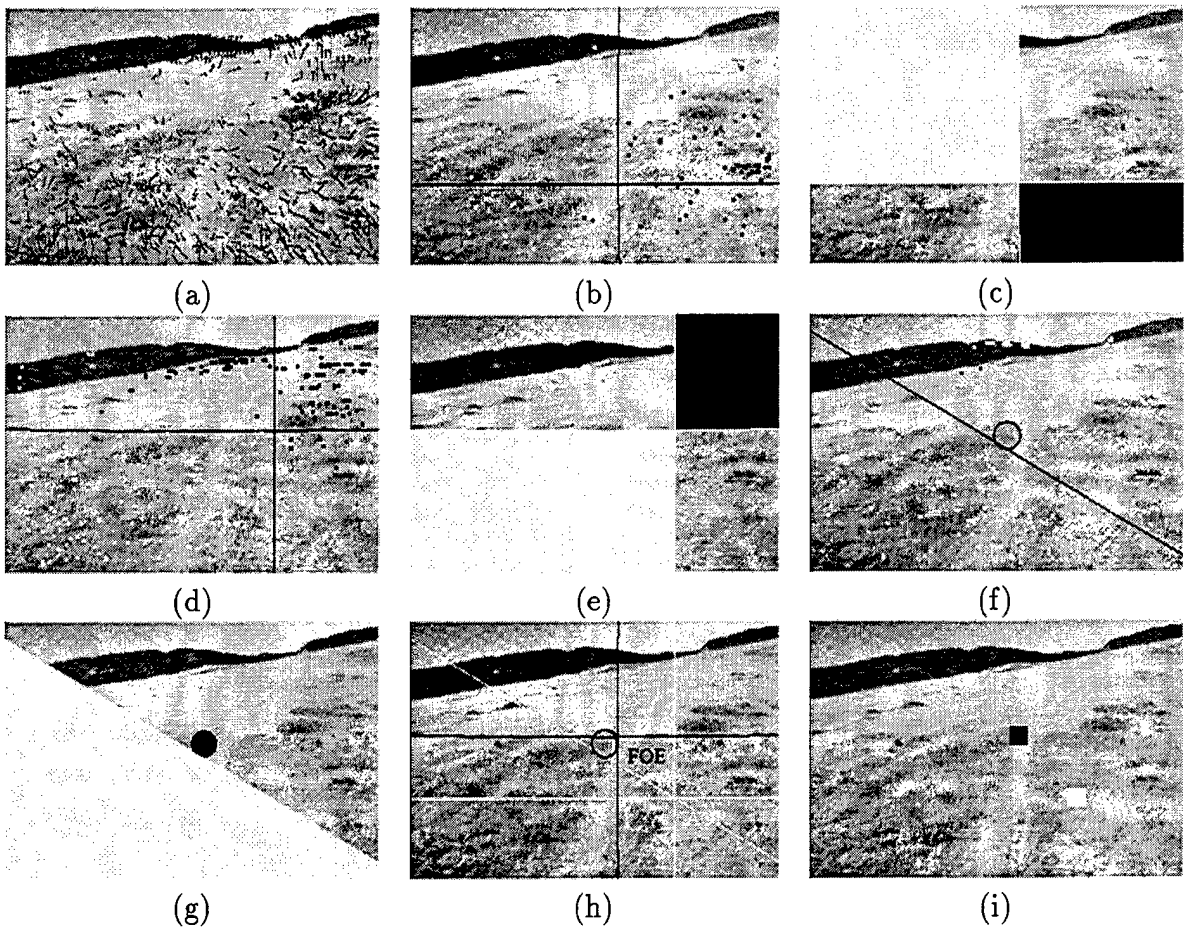
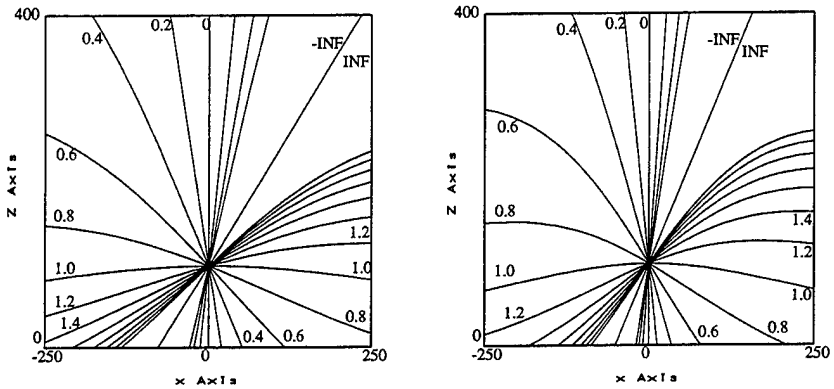


Figure 14: A camera mounted on the captured a sequence of images as the vehicle moved along rough terrain in the countryside, thus undergoing continuously changing rigid motion. (a) shows one frame of the sequence with the normal flow field overlaid. (b), (d) and (f) show the positive (light color) and negative (dark color) vectors of the longitudinal patterns corresponding to the x-, y- and z-axes (see [4]). (c), (e) and (g) show the corresponding fitted patterns. (h) shows, superimposed on the image, the boundaries of the patterns whose intersections provide the FOE and the AOR (the point where the rotation axis pierces the image plane). (i) Measurements are not everywhere available (strong intensity gradients are sparse), but a set of patterns can still be fitted, resulting in two bounded areas as locations for the FOE and the AOR.

Gaussian derivative spatial filters at a number of orientations and scales. Such feature vectors serve as effective photometric descriptions of the local intensity variations present in the image region about a scene/object point; in addition, they can be made rotation and scale invariant [16]. The iconic feature vectors are stored in two separate memories. One memory is indexed by image coordinates while the other is indexed by object coordinates. Object



(a) FOV=50° (b) FOV=70°

Figure 15: Iso-distortion contours resulting from intersecting the iso-distortion surfaces with the Zx-plane (the plane defined by the optical axis and the horizontal axis of the camera). The horizontal component x_0 of the actual Focus of Expansion is $x_0 = 50$. Assuming that the error in estimating x_0 is $x_{0e} = -50$ and the error in estimating the rotation around the y-axis is $\beta_e \simeq 0.001$, Figures 15a and 15b show iso-distortion contours for two different values of the FOV. The value next to each contour denotes the amount of multiplicative distortion (1.0 means no distortion).

location matches a localized set of model features with image features at all possible retinal locations. Object identification matches a foveal set of image features with all possible model features.

We describe here in more detail the routine for object location; details regarding the identification routine, which employs Kalman Filter theory and visual learning, can be found in [15]. The location routine crucially depends on the fact that only a single model object is being matched to other objects in an image at any instant. Let us denote this model that is to be located in the current image as

$$M = \{\mathbf{r}^m, m = 1, \ldots, m_{\max}\}. \tag{20}$$

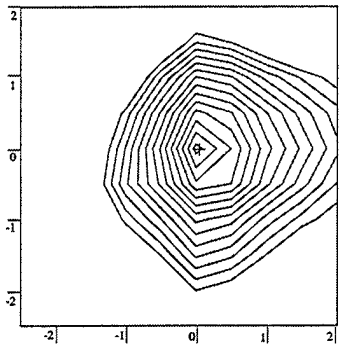
where $\mathbf{r}^m$ are the object's filter response vectors extracted from different spatial locations.

The location algorithm in its most general form proceeds as follows:

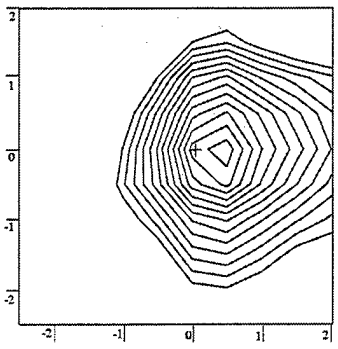
1. For each response vector $\mathbf{r}^m$ representing some model point m, create a *Saliency Image* S_m defined by

$$S_m(x, y) = \|\mathbf{r}(x, y) - \mathbf{r}^m\|^2 \tag{21}$$

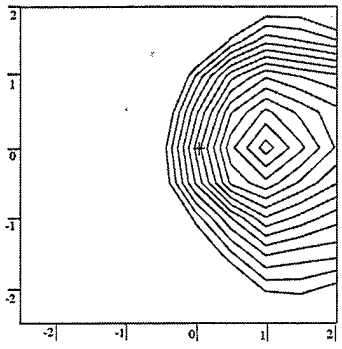
(a) Indoor scene 5 m away



(b) $\left(\frac{Z}{W}\right)_{\text{mid}} < 2.5\ s^{-1}$



(c) $\left(\frac{Z}{W}\right)_{\text{mid}} < 5\ s^{-1}$



(d) $\left(\frac{Z}{W}\right)_{\text{mid}} < 10\ s^{-1}$

Figure 16: The effectiveness of a relatively inexpensive, not highly accurate inertial sensor depends on the distribution of depths in the scene in view. The analysis using iso-distortion contours is based on whether "negative" depth values arise, and considers as a criterion for the estimation of the FOE the point that gives rise to a minimum number of non-positive depth measurements. The level contours in (b), (c), and (d) show the variation in the number of negative depths as the FOE estimates move away from the true FOE (indicated by the cross). The best FOE estimate is associated with the "bottom" of the contours (minimum number of negative depths). The axes of these contour plots represent the error of the FOE in degrees; they are not plotted at the same scale as the image in (a). FOE = 30°; $\beta_e = 0.04°/s$ (error in rotation around the y-axis); $(Z/W)_{\text{mid}}$ adjusted by changing W. The analysis suggests that an inertial sensor with an accuracy of $0.04°/s$ may be problematic in outdoor scenes but should be very successful in indoor scenes.

2. Find the best match point in the image for each m using the following Winner-Take-All rule:

$$(x_{b_m}, y_{b_m}) = \text{argmin}_{(x,y)}\{S_m(x,y)\} \tag{22}$$

3. Construct a binary image B:

$$B(x,y) = \begin{cases} 1 \text{ if } (x,y) \in \{(x_{b_m}, y_{b_m})\} \\ 0 \text{ otherwise} \end{cases} \tag{23}$$

where $m = 1, \ldots, m_{\max}$.

4. Output the location of the object in the current image as

$$(x_b, y_b) = \text{argmax}_{(x,y)} \{S(x,y) * B(x,y)\} \tag{24}$$

where B is an appropriate blurring function whose size can usually be estimated in an active vision environment.

The location algorithm operates at close to real-time rates in an active vision system consisting of the University of Rochester binocular head with two movable color CCD cameras that provide input to a Datacube MaxVideo^{TM} MV200 pipeline image-processing system. Given a live input image (of size 512×480) from the camera, the MV200 executes nine convolutions using nine different 8×8 discrete Gaussian derivative filter kernels on a low-pass filtered five-level pyramid of the image to obtain the response vectors for all points in the current image; these vectors are stored in a "memory surface" $\mathcal{S}$. During the memorization phase, filter responses are extracted for each of the sparse set of points located within the given object. During the location phase, a model response vector is loaded into the 8×8 convolution kernel and convolved with the memory surface $\mathcal{S}$ containing the response vectors for each point of the input image; the closest vectors can be selected by simply thresholding the results of the convolution at individual thresholds to obtain candidate match points.

Figure 17 shows an example of the performance of the location routine in a realistic scene. Here we demonstrate the algorithm's ability to find a model object (in this case, the stuffed doll) in the presence of object motion, clutter, and perspective distortion; '+' denotes the best matching location found by the algorithm.

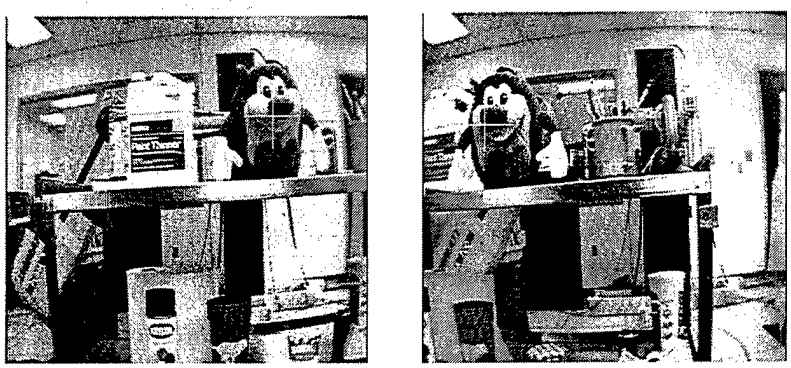


Figure 17: Example of locating a model object (stuffed doll) under conditions of motion, clutter, and perspective.

3.5 Active Intelligent Observers—University of Pennsylvania

Current active vision systems address two primary questions: how to select interesting parts of the scene to look at and how to maintain acquisition of the selected objects. We are interested in building an active intelligent observer on top of a reflexive gaze control system. Specifically, we will use high-level knowledge to direct the actions of an active vision system using feedback from low-level gaze control mechanisms.

Our approach comprises three phases. In the teaching phase, the active intelligent observer acquires a series of views of a reference object and integrates them into a structural model of the object. In the acquisition phase, the observer searches for the desired object in the scene and establishes the correct image size by moving and zooming. In the guidance phase, the observer dynamically constructs a gaze control path that leads to the optimal aspect for the current task. The structural model of the object allows the observer to determine the location of the optimal aspect in the view sphere and to generate intermediate views that guide the observer along the gaze control path. In robotic applications, the manipulated object must frequently be examined as to its identity and orientation. The active intelligent observer uses the structural model to determine the object's orientation and to move around it to view specific features.

Low-level image measures for gaze control are notoriously sensitive to changes in scale, orientation, and viewing aspect. However, if a simple template is augmented with high-level structural information, new views can be synthesized to guide the observer's gaze between

known views. Conversely, the observer can infer the pose of an object from the current view by comparing it to the stored knowledge. The basic idea of this approach is to add a higher level of feedback to gaze control and close the perception-action loop around the visual servoing task.

4 Acknowledgments

Contributions to this report were made by Carlos Morimoto, Center for Automation Research, University of Maryland, College Park, MD; Martin Herman, Intelligent Systems Dvision, National Institute of Standards and Technology, Gaithersburg, MD; Ruzena Bajcsy, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA; and Randal Nelson, Department of Computer Science, University of Rochester, Rochester, NY.

5 List of Publications and Technical Reports

5.1 Publications

L.S. Davis, R. Bajcsy, M. Herman, and R. Nelson, RSTA on the Move, Proc. ARPA Image Understanding Workshop, Monterey, CA, November 13-16, 1994, 435-456.

L.S. Davis, R. Bajcsy, M. Herman, and R. Nelson, RSTA on the Move: Detection and Tracking of Moving Objects from an Automonomous Mobile Platform, Proc. ARPA Image Understanding Workshop, Palm Springs, CA, February 12-15, 1996, 651-664.

5.2 Technical Reports

Estimation of Vehicle Dynamics from Monocular Noisy Images, Y. Yao and R. Chellappa, CS-TR-3172, November 1993.

Tracking a Dynamic Set of Feature Points, Y. Yao and R. Chellappa, CS-TR-3294, June 1994.

Estimation of Unstabilized Components in Vehicular Motion, Y. Yao and R. Chellappa; CS-TR-3349, September 1994.

Interaction Between 3-D Shape and Motion: Theory and Applications, L. Cheong, C. Fermüller, and Y. Aloimonos, CS-TR-3480, May 1995.

3-D Model-Based Image Stabilization Using Multiple Visual Cues, Y. Yao, P. Burlina, R. Chellappa, and T.Y Wu, CS-TR-3506, July 1995.

Off-Road Navigation from Selective Stabilization, Y. Yao and R. Chellappa, CS-TR-3509, August 1995.

Electronic Stabilization and Feature Tracking in Long Image Sequences, Y. Yao, CS-TR-3527, September 1995.

Performance Characterization of Image Stabilization Algorithms, S. Balakirsky and R. Chellappa, CS-TR-3630, April 1996.

Explaining Human Visual Space Distortion, C. Fermüller, L. Cheong, and Y. Aloimonos, CS-TR-3662, July 1996.

6 List of Participating Scientific Personnel

Yiannis Aloimonos

Stephen B. Balakirsky (M.S., 1995)

Rama Chellappa

Loong Fah Cheong (Ph.D., 1996)

Larry S. Davis

Daniel DeMenthon

Sara Larson

Carlos Morimoto

Yi-Sheng Yao (Ph.D., 1995)

7 Report of Inventions

NONE

8 Bibliography

[1] S. Balakirsky and R. Chellappa. Performance characterization of image stabilization algorithms, Center for Automation Research Technical Report, CAR-TR-822, University of Maryland, College Park, MD, April 1996.

[2] P. Burt and P. Anandan. Image stabilization by registration to a reference mosaic. In *Proc. of ARPA Image Understanding Workshop,* Monterey, CA, November 1994, pp. 425–434.

[3] L. Cheong and Y. Aloimonos. Isodistortion contours and egomotion estimation. In *Proc. of IEEE International Symposium on Computer Vision*, Coral Gables, FL, November 1995, pp. 70–76.

[4] C. Ferm üller and Y. Aloimonos. Qualitative egomotion. *International Journal of Computer Vision*, 15:7–29, 1995.

[5] M. Hansen, P. Anandan, K. Dana, G. van der Wal, and P.J. Burt. Real-time scene stabilization and mosaic construction. In *Proc. of ARPA Image Understanding Workshop,* pages 457–465, Monterey, CA, November 1994, pp. 457–465.

[6] K. Hosoda, H. Moriyama, and M. Asada. Visual servoing utilizing zoom mechanism. In *Proc. of IEEE International Conference on Robotics and Automation*, Nagoya, Japan, May 1995, pp. 178–183.

[7] J. Hwang, Y. Ooi, and S. Ozawa. An adaptive sensing system with tracking and zooming a moving object. *IEICE Transactions on Information and Systems*, E76-D:926–934, 1993.

[8] B.C. Madden and U.M. Cahn von Seelen. PennEyes: A binocular active vision system. Technical Report, GRASP Laboratory, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, to appear.

[9] C.H. Morimoto, D. DeMenthon, L. Davis, R. Chellappa, and R. Nelson. Detection of independently moving objects in passive video. In *Proc. of IEEE Intelligent Vehicles Symposium*, Detroit, MI, pp. 270–275, 1995.

[10] R.C. Nelson. Qualitative detection of motion by a moving observer. *International Journal of Computer Vision* 7:33–46, 1991.

[11] R.C. Nelson and R. Polana. Qualitative recognition of motion from temporal texture. *CVGIP: Image Understanding*, 56:78–89, 1992.

[12] H.S. Parry, A.D. Marshall, and K.C. Markham. Integration of segmentation information and correlation technique for tracking objects in sequences of images. In *Proc. of Videometrics IV (SPIE Proceedings vol. 2598)*, Philadelphia, PA, October 1995, pp. 208–219.

[13] R. Polana and R.C. Nelson. Detecting activities. *Journal of Visual Communication and Image Representation*, 5:172–180, 1994.

[14] R. Polana and R.C. Nelson. Detecting activities. In *Proc. of ARPA Image Understanding Workshop*, Washington, DC, April 1993, pp. 569–574.

[15] R.P.N. Rao and D.H. Ballard. Dynamic model of visual memory predicts neural response properties in the visual cortex. Technical Report 95-1, National Resource Laboratory for the Study of Brain and Behavior, 1995.

[16] R.P.N. Rao and D.H. Ballard. An active vision architecture based on iconic representations. *Artificial Intelligence* 78:461–505, 1995.

[17] I.D. Reid and D.W. Murray. Tracking foveated corner clusters using affine structure. In *Proc. of International Conference on Computer Vision*, Berlin, Germany, May 1993, pp. 76–83.

[18] Q. Tian and M.N. Huhns. Algorithms for subpixel registration. *Computer Vision, Graphics, and Image Processing*, 35:220–233, 1986.

[19] Y.S. Yao, P. Burlina, and R. Chellappa. Stabilization of images acquired by unmanned ground vehicles. In *Proc. of ARPA Image Understanding Workshop*, Palm Springs, CA, February 1996, pp. 687–694.

[20] Q. Zheng and R. Chellappa. A computational vision approach to image registration. *IEEE Transactions on Image Processing* 2:311–326, 1993.

[21] Q. Zheng and R. Chellappa. Automatic feature point extraction and tracking in image sequences for unknown camera motion. *International Journal of Computer Vision*, 15:31–76, 1995.